

1.

A control system is used to control sliding doors which automatically open to allow people in and out of a shop.

- (a) (i) Identify **one** type of sensor in this system. [1]
  - (ii) Identify **one** piece of hardware, other than sensors, that is part of the control system. [1]
  - (iii) With reference to the role of sensors, outline the sequence of steps within the computer control system that will take place when a person approaches the door. [3]
  - (b) (i) Define the term *interrupt*. [2]
  - (ii) Describe a situation in this system where an interrupt would occur. [2]
  - (c) Discuss the contribution of computer control systems in industry where they replace human workers. [6]
- 
- (a) (i) *Award up to [1 max].*  
Proximity;  
Movement;  
Pressure; [1]
  - (ii) *Award up to [1 max].*  
Transducers;  
AD converters;  
Actuators;  
Micro-processor; [1]
  - (iii) *Award up to [3 max].*  
When a person approaches, sensors activate;  
Signal sent to processor;  
Which sends signal to actuator/transducer (which opens doors);  
After fixed time/no further sensory input, doors close; [3]
- 
- (b) (i) *Award up to [2 max].*  
Interrupt is a signal sent to the processor;  
Sent by hardware or software;  
Indicating an event that needs the processor's immediate attention; [2]
  - (ii) If a second person approaches the door while it is closing;  
This will interrupt the processing cycle and the door will re-open; [2]

- (c) Award **[1]** for an advantage/disadvantage and **[1]** for an expansion, for 3 examples, up to **[6 max]**.

*Example discussion points:*

Labour cost;  
Quality of work;  
Retraining;  
Redundancy;  
Performance (of repetitive tasks);  
Productivity;  
Safety;

*Example answer:*

Initially a computer system is more expensive;  
Once the computer control system is installed/set up it is more economical;  
(Over longer period of time), than human labour;  
Computers can work accurately;  
7 days/24 hours;  
Performing monotonous/unpleasant tasks without complaining;  
In dangerous conditions (fumes, poison, lifting heavy weight, etc);

**[6]**

2.

The table below holds student names and scores, from a class test.

NAME	SCORE
Ann Taylor	10
Boris Penn	18
Ivan Troth	8
Peter Hu	9
Mary Looty	7

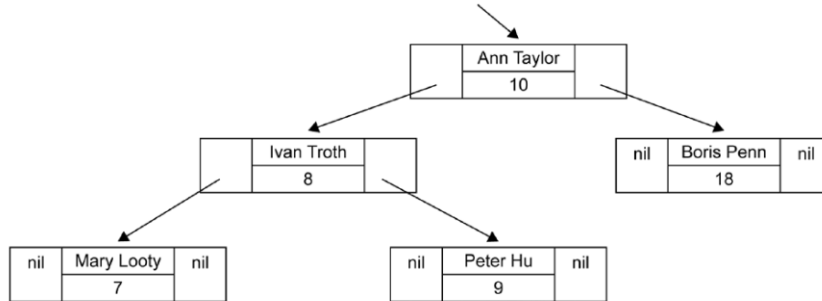
- (a) Draw a diagram to show how the data given in the table could be stored in a binary tree in the order of scores. Data should be inserted into the binary tree in the order given in the table (*ie* data about Ann Taylor is to be inserted first).

**[3]**

- (b) The same data could be inserted into a singly linked list in descending order of scores. Draw a diagram of this singly linked list.

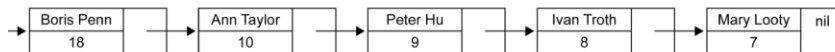
**[3]**

- (a) Award marks as follows up to **[3 max]**.  
 Award **[1]** for correct nodes (2 data fields and two pointers).  
 Award **[1]** for correct root.  
 Award **[1]** for correct left subtree.  
 Award **[1]** for correct right subtree.  
 Award **[1]** for trees that display either only names or only numbers, provided that they are structurally correct.



**[3]**

- (b) Award marks as follows up to **[3 max]**.  
 Award **[1]** for correct nodes (2 data fields and one pointer field).  
 Award **[1]** for correct order (links shown).  
 Award **[1]** for the external pointer to list and showing the end of the list (nil).



**[3]**

- (c) Compare the data structures in part (a) and part (b) in terms of:

(i) searching [2]

(ii) storage requirements. [2]

- (d) Consider the following **recursive** algorithm, in which  $x$  and  $y$  are parameters in the method  $F$ . The **return** statement gives the value that the method generates.

```

F(X, Y)
  if X < Y then
    return F(X+1, Y-2)
  else if X = Y
    return 2 * F(X+2, Y-2) - 2
  else
    return 2 * X + 4 * Y
  end if
  
```

Determine the value of  $F(5, 11)$ .

**[5]**

- (c) (i) Award **[2]** for a valid comparison of binary trees and linked lists.  
Award **[1]** for identifying a characteristic of either binary trees or linked lists, but without an explicit comparison.

Example answer

Searching:

<b>Binary tree</b>	<b>Singly linked list</b>
Binary	Linear (sequential)
Faster/efficient	Slower/less efficient

**[2]**

- (ii) Award **[2]** for a valid comparison of binary trees and linked lists.  
Award **[1]** for identifying a characteristic of either binary trees or linked lists, but without an explicit comparison.

Example answer

Storage requirements:

<b>Binary tree</b>	<b>Singly linked list</b>
More storage needed	Less storage
Requires two pointers	Requires only one pointer

**[2]**

- (d) Award **[1]** for each correct line.

$$\begin{aligned}
 F(5, 11) &= F(6, 9) \\
 &= F(7, 7) \\
 &= 2 * F(9, 5) - 2 \\
 &= 2 * (2 * 9 + 4 * 5) - 2 \\
 &= 2 * 38 - 2 = 74
 \end{aligned}$$

**[5]**

3.

Describe how a GPS system can identify the position of a person.

**[3]**

Award up to **[3 marks max]**.

GPS works by communication with satellites;

By knowing the position of the satellite (sent to GPS device);

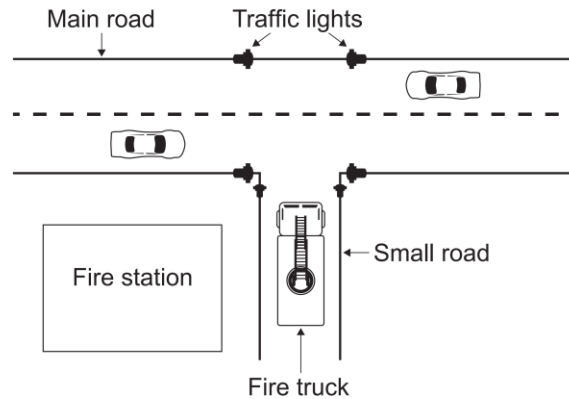
And calculating the time difference between satellites;

The position of the device can be calculated;

**[3]**

4.

In a town, a set of traffic lights control access from a small road, where a fire station is located, to a main road that has heavy traffic. In times of emergency, many vehicles from the fire station may need to leave the station at the same time. A system is put in place so that when a fire truck on the small road approaches the main road, the traffic lights switch to green (Go) on the small road and to red (Stop) on the main road.



- (a) Outline the role of sensors and a microprocessor in controlling the traffic lights in this way. [4]
- (b) Suggest how the traffic lights can be changed back to their original state once there are no more fire trucks coming from the small road. [3]

These traffic lights are controlled by embedded systems at the point of use. It is proposed that they should be controlled from the same central computer as all the other traffic lights in the town.

- (c) Discuss the **advantages** and **disadvantages** of running the town's traffic light system on one central computer with multiple inputs and outputs. [5]

A series of cameras are installed at each of the town's traffic lights. These cameras are connected to the central computer.

- (d) Discuss the social implications of monitoring traffic in this way. [3]
- (a) Sensors will be used to detect the approach of a vehicle from the minor road;  
Likely to be touch/weight sensor embedded in the road;  
Sensor input is converted from analog to digital;  
To be processed and;  
Signal sent to switch traffic lights; [4]
  - (b) *Award up to [3 marks max].*  
Continual feedback from sensor to processor;  
A calculation based on number of vehicles/speed/etc or time taken for a vehicle to pass;  
(Timer) resets if another vehicle is detected;  
Once no input for a certain time traffic lights changed back; [3]

- (c) Award **[2 marks]** for advantages, **[2 marks]** for disadvantages and **[1 mark]** for weighing up.

**Disadvantages:**

Central computer would have to cope with inputs from many places;  
With differing priorities which could take time;  
Connection failure possible from a particular point;  
Computer failure puts all lights in the area out;  
Cost of communication system/central control system;

**Advantages:**

More control over traffic flow at these points;  
Lights can be adapted from distance to avoid traffic blocks;  
Any problem appearing at one point is known immediately and can be dealt with;  
Cheaper as no need for communication software/hardware/control centre;  
Can react/change rules to changing levels of traffic flow;

Overall, it would be better to ... (*appropriate conclusion*);

**[5]**

- (d) Award **[1 mark]** for an advantage outlined, **[1 mark]** for a disadvantage outlined and **[1 mark]** for discussing.

*For example:*

Controlling the movement of vehicles and identifying people who speed should help to reduce accidents (as motorists know that they will be caught if driving dangerously);  
This could also save lives;

Individual displacement is tracked;  
Which can be seen as an infringement on personal liberty/a breach of privacy;  
In some cases the information could be used unjustly against the individual (eg in times of political unrest);

It comes down to physical safety on the road against privacy/personal liberty;

**[3]**

## **Part 2**

### **Option D — Object-oriented programming**

A small health clinic with three doctors operates in a village. All clients of the clinic have their details stored in the clinic's database. Patients that visit the clinic during the day are given a priority rating (1–3) and are seated in a waiting room to wait for the next available doctor. When it is their turn, the patients are taken from the waiting room to have a consultation with their assigned doctor, who makes a diagnosis, provides treatment and writes a prescription.

The clinic's system is coded in Java. There are many objects in this system and some of them are listed below.

Object	Description
Doctor	A licensed professional who treats patients in the clinic.
Patient	A sick person who requires a consultation with a doctor.
WaitingRoom	A place where patients wait for their consultations.
Consultation	A dated meeting between a doctor and a patient which results in a diagnosis, treatment and a prescription for medication.
Treatment	A dated record of all actions and medication prescribed to treat the patient's diagnosed condition.

The three objects `Patient`, `WaitingRoom` and `Treatment` have been defined in the following UML diagrams:

Patient
Integer id String name Integer priority String doctor
setId (Integer id) setName (String name) setPriority (Integer priority) setDoctor (String doctor) Integer getID() String getName() Integer getPriority() String getDoctor() String toString()

WaitingRoom
Patient[10]patients
add(Patient newPatient) void callNextPatient() Integer findNextPatientIndex() remove(Integer n)

Treatment
String date Integer patientId String doctor String actions String medication
setDate (String date) setPatientId (Integer id) setDoctor (String doctor) setActions (String actions) setMedication (String medication) String getDate() Integer getPatientID() String getDoctor() String getActions() String getMedication() String toString()

The Patient and WaitingRoom objects are implemented as follows:

```
public class Patient
{
    private int id;
    private String name;
    private int priority;
    private String doctor;

    public Patient(int i, String n, int p)
    {
        id = i;
        name = n;
        priority = p;
        doctor = null;
    }
    public void setId(int i) { id = i; }
    public void setName(String n) { name = n; }
    public void setPriority(int p) { priority = p; }
    public void setDoctor(String d) { doctor = d; }
    public int getId() { return id; }
    public String getName() { return name; }
    public int getPriority() { return priority; }
    public String getDoctor() { return doctor; }
    public String toString() { return id+" "+name+" "+priority+" "+doctor; }
}

public class WaitingRoom
{
    private Patient[] patients = new Patients[10];

    // uses default constructor

    public void add(Patient newPatient)
    // adds the new patient in the next empty array location
    {
        int i = 0;
        while ((patients[i] != null) && (i < 10))
        {
            i=i+1;
        }
        if (i==10) { System.out.println("No more space in the waiting room."); }
        else { patients[i] = newPatient; }
    }
}
```



```

public void callNextPatient()
// finds the next patient, outputs their details
// and removes the patient from the array
{
    int index = 0;
    if (patients[0]==null)
    {
        System.out.println("The waiting room is empty.");
    }
    else
    {
        index = findNextPatientIndex();
        remove(index);
    }
}

private int findNextPatientIndex()
// returns the index of the first patient with the
// highest priority in the array patients
{
    int max = 0;
    //... code missing ...
    return max;
}

private void remove(int n)
// outputs the data of the patient instance at array index n
// and removes that patient by shifting all remaining patients
// by one index towards the front of the array
{
    //... code missing ...
}
}

```

5. ( no need to answer, just for your refrence)

- (a) Define the term *constructor*, using an example from the code on pages 14 and 15. [2]
- (b) Describe **one** additional field that might have been included in the `Patient` class. Include a data type and sample data in your answer. [2]
- (c) Describe the relationship between the `Patient` object and the `WaitingRoom` object. [2]

Consider the `WaitingRoom` class as presented on pages 14 and 15.

- (d) Construct the missing lines of code in the `findNextPatientIndex()` method to return the index of the first patient with the highest priority in the `patients` array. **Note:** the highest possible priority is 3. [3]
- (e) Construct the `remove(int n)` method which outputs the data of the patient object at index `n` and then removes that patient object by moving all remaining patient objects one index towards the front of the `patients` array. You may assume that `n` is a valid index between 0 and 9, and that an instance of `Patient` exists at that index. [6]

6. . ( no need to answer, just for your reference)

- (a) In relation to the `Patient` class, outline **one** advantage of encapsulation. [2]
- (b) In relation to the `Treatment` object, discuss **one** ethical consideration when designing software that stores patients and their illnesses. [4]

The clinic would like to start storing details in a `Doctor` object, including full name, telephone number and whether the doctor is present or not. For example:

name:	Dr Henriëtte Mănescu-Rața
phone:	0734511122
present:	true

- (c) Design the `Doctor` object using a UML diagram. [3]
- (d) In relation to the `Doctor` object, outline the need for extended character sets as used by modern programming languages. [3]

7. . ( no need to answer, just for your reference)

`Treatment` objects are being instantiated throughout the day and added to a collection. The object `treatmentFile` contains the following methods which act on that collection:

- `getNext()` which reads the next treatment from the collection and returns it
- `hasNext()` which returns false when there are no more treatments in the collection.

Construct the method `showMedicationByDoctor()`, which will take the name of a doctor as a parameter and output the medication for each treatment in the collection that has been provided by that doctor. You may assume that `treatmentFile` has been declared as a global variable, that it is open for reading, and that the first time `getNext()` is called it will return the first treatment from the collection. [6]

8. ( Part 2 question starts from here, please answer this!)

The `Treatment` object needs to be developed further. There are three possible types of treatment and this is to now be recorded.

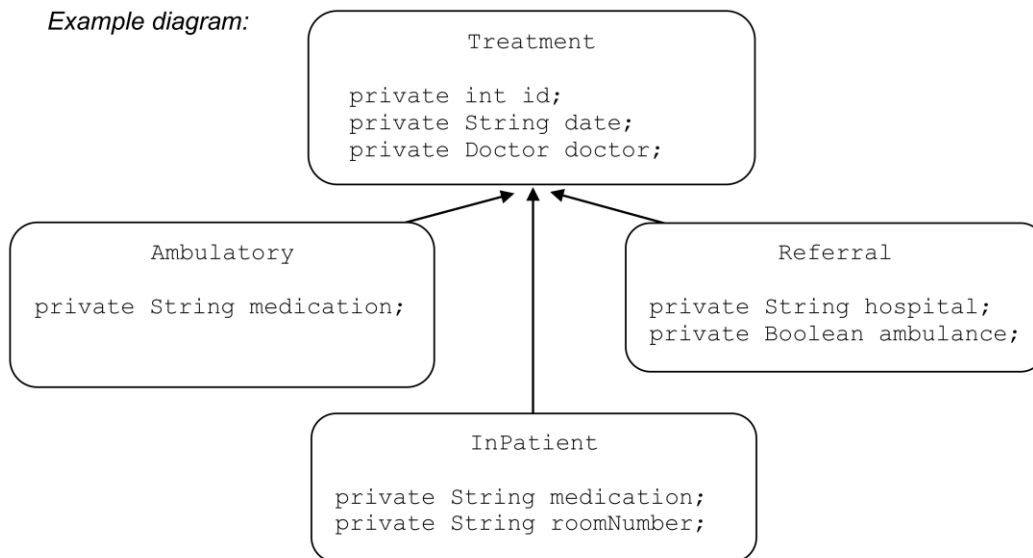
- `ambulatory` – the patient is treated and goes home afterwards
- `in-patient` – the patient spends one or more nights in the clinic
- `referral` – the patient is sent to a hospital in a nearby city.

All treatments have common fields such as ID of the patient, date and a doctor object, but other fields are different. For example, `ambulatory` and `in-patient` treatments include medication while `referral` does not. On the other hand, `referral` includes the name of the hospital that the patient was sent to and whether or not ambulance transportation was used. `In-patient` treatment includes a room number.

- (a) Construct diagrams to show how inheritance can be used to re-design the `Treatment` class. [6]
- (b) Describe **three** advantages of modularity in program development. [6]

- (a) Award marks as follows up to **[6 max]**.  
 Award **[1]** for a common super-class.  
 Award **[1]** for all common fields: *id* field (integer) *date* field (String) *doctor* field (String).  
 Award **[1]** for three sub-classes and award **[1]** for arrows with correct direction.  
 Award **[1]** for the *medication* fields in *Ambulatory* and *InPatient*, but not in *Referral*.  
 Award **[1]** for the additional fields (*Hospital* and *Ambulance*) in *Referral*.  
 Award **[1]** for the additional fields (*RoomNumber* and *ReleaseDate*) in *InPatient*.

Example diagram:



Do not penalize if data types are not specified.

**[6]**

- (b) Award **[1]** for identifying each advantage and an additional **[1]** for an elaboration of this advantage, up to **[6 max]**.

Example answer:

Faster development;  
 Because different programming teams can work on different modules;

Easier to debug;  
 Because the smaller modules will have fewer mistakes than one big program;

Easier to update (in the future);  
 Because it is easier to update a module than the full program;

Re-usability;  
 Modules can be stored in libraries and reused for different programs;

**[6]**

9.

Due to the growth of the village, more people have been using the clinic and the current static implementation of the waiting room is no longer suitable. The waiting room needs to be implemented dynamically with a structure that preserves the order that the patients have come into the clinic.

- (a) (i) State why a stack is not suitable for this purpose.

**[1]**

- (ii) Outline **one** typical application of a stack.

**[2]**

- (a) (i) A stack is a LIFO structure; [1]
- (ii) Award [1] for stating an application and [1] for an elaboration.  
*Example answer:*  
 The system stack stores return data of interrupted processes;  
 The last interrupted process is the first to resume; [2]

It has been decided that a linked-list will be used to hold the individual `Patient` objects.

An object of the `LinkedList` class will be used to instantiate a dynamic list of patients to implement the `WaitingRoom` as follows.

```
public class WaitingRoom
{
    private LinkedList<Patient> PatientList = new LinkedList<Patient>();
    // methods
    public void add(Patient P)
    // adds a patient at the end of the list
    {
        PatientList.addLast(P);
    }
    public void remove()
    // outputs the name of the next patient to see a doctor and
    // removes this patient instance from the list
    {
        int index = findNextPatientIndex();
        System.out.println(PatientList.get(index).getName());
        PatientList.remove(index);
    }
    ,
    private int findNextPatientIndex()
    {
        int i = 0, result = 0;
        Patient current, firstup;
        firstup = new Patient();
        firstup.setPriority(0);
        while (i < PatientList.size())
        {
            current = PatientList.get(i);
            if (current.getPriority() > firstup.getPriority())
            {
                firstup = current;
                result = i;
            }
            i=i+1;
        }
        return result;
    }
}
```

- (b) The `remove` method could cause a run-time error. State the pre-condition for the `findNextPatientIndex` method, in order to avoid this error. [1]
- (b) Pre-condition: there is at least one patient instance in the list; [1]

- (c) Consider the following list:



**Copy** and complete the table below to trace a call to the `findNextPatientIndex` method for this list. **Note:** the initialization is given in the first row.

i	current.name	firstup.name	result
0	–	null	0
0			

[3]

- (c) Award **[1]** for each row filled correctly.

i	current.name	firstup.name	result
0	-	null	0
0	Abdul Hashim	Abdul Hashim	0
1	Iris Gotenberg	Abdul Hashim	0
2	Anh Nguyen	Anh Nguyen	2

[3]

- (d) State the purpose of the `findNextPatientIndex` method.

[1]

- (e) Outline the changes needed to improve the `findNextPatientIndex` method with an early exit from the loop.

[3]

- (d) `findNextPatientIndex` returns the index of the first patient with the highest priority in the list/find the next patient with the highest priority;

[1]

- (e) Award marks as follows up to **[3 max]**.

Award **[1]** for introducing and initializing a Boolean `done`.

Award **[1]** for adding `!done` to the loop condition.

Award **[1]** for testing for early exit (`current.priority==3`).

*Example answers:*

Add a boolean `done` and set it to false before the loop;

Add `&&(!done)` to the loop condition;

if `current.priority` equals 3 then make `done` true;

```

boolean done = false;
while ((i < PatientList.size()) && (!done))
{
    current = PatientList.get(i);
    if (current.getPriority() > firstup.getPriority())
    {
        firstup = current;
        result = i;
    }
    if (current.getPriority()==3) { done = true; }
    i=i+1;
}
  
```

```

while ((i < PatientList.size()) && (current.getPriority()!=3))
  
```

[3]

10.

Consider the following fragment of code that implements the method `result()`, where `x` and `y` are non-negative numbers.

```
public int result(int x, int y)
{
    if (x==0)
    {
        return 0;
    }
    else if (y == 0)
    {
        return 1;
    }
    else
    {
        return (x+y)*result(x, y-1)
    }
}
```

(a) Define *recursion*. [1]

(b) Trace the method `result(3, 4)` showing the intermediate steps and the final evaluation of this call. [4]

(a) A programming technique where a method calls on itself; [1]

(b) Award up to **[4 max]**.  
Award **[1]** for correct result, 840.  
Award **[1]** for applying base case to `result(3,0) =1`.  
Award **[1]** for showing at least 2 recursive calls of `result()`, even if the final calculation is incorrect.  
Award **[1]** for presentation, showing the alternation of `+` and `*`.

```
result(3,4)
(3+4)* result (3,3)
(3+4)* (3+3)* result (3,2)
(3+4)* (3+3)*(3+2)* result (3,1)
(3+4)* (3+3)*(3+2)*(3+1)* result (3,0)
(3+4)* (3+3)*(3+2)*(3+1)*1 = 840
```

[4]

An electronic amplifier uses a variant of `result()`, to produce distorted special effects whilst a music instrument is being played.

The method `result4()`, with the signature

```
double result4(double x, int y, double z, int v),
```

produces these special effects, taking four non-negative signals in input, and obeying the following specification:

- if the first three inputs ( $x$ ,  $y$  and  $z$ ) have a combined value that is more than 12 times the value of  $v$ , then the method behaves as `result(v, y)`
- otherwise, and if the value of  $y$  is less than half the value of  $x$ , the method `result4()` is called again, only now with  $y$  and  $v$  decremented by 3, unless this would make their values invalid
- if none of these actions are possible, `result4()` behaves as `result(0, 0)`.

(c) Construct the method `result4()`, according to the given specification.

[4]

(c) **Award up to [4 max]**

*Award [1] for having a return call for each branch*

*Award [1] for both cases returning a call to `result()`.*

*Award [1] for checking the condition on  $x$ ,  $y$  being non-negative.*

*Award [1] for correct recursive call on `result4()`.*

```
public double result4(double x, int y, double z, int v)
{
    double sum = x + y + z;
    if (sum > 12*v)
    {
        return result(v,y);
    }
    else
    {
        if ((y<x/2)&& (v-3>=0 && y-3 >=0))
        {
            return result4(x,y-3,z,v-3);
        }
        else
        {
            return result(0,0);
        }
    }
}
```

[4]